#2

Docket No. 1095.1184

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:

Katsumi ICHINOSE, et al.

Serial No.: NEW

Filed: April 19, 2001

For: **INFORMATION PROCESSING METHOD AND RECORDING MEDIUM**

Group Art Unit: Unassigned

Examiner: Unassigned

SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN APPLICATION IN ACCORDANCE WITH THE REQUIREMENTS OF 37 C.F.R. §1.55

Honorable Commissioner of Patents and Trademarks
Washington, D.C. 2023l

Sir:

In accordance with the provisions of 37 C.F.R. §1.55, the applicant(s) submit(s) herewith a certified copy of the following foreign application:

Japanese Patent Application No. 2000-360397, filed: November 28, 2000.

It is respectfully requested that the applicants be given the benefit of the foreign filing date as evidenced by the certified papers attached hereto, in accordance with the requirements of 35 U.S.C. §119.

Respectfully submitted,

STAAS & HALSEY LLP

Date: April 19, 2001

By: _____

James D. Halsey, Jr.
Registration No. 22,729

700 11th Street, N.W., Ste. 500
Washington, D.C. 20001
(202) 434-1500

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

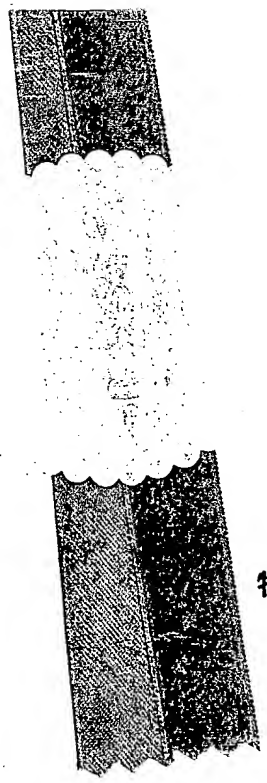
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: ２０００年１１月２８日

出 願 番 号
Application Number: 特願２０００－３６０３９７

出 願 人
Applicant (s): 富士通株式会社

CERTIFIED COPY OF
PRIORITY DOCUMENT



２００１年 ２月２３日

特 許 庁 長 官
Commissioner,
Patent Office 及 川 耕 造

出証番号 出証特２００１－３０１１３４４

【書類名】　特許願
【整理番号】　0050916
【提出日】　平成12年11月28日
【あて先】　特許庁長官殿
【国際特許分類】　G06F　9/46
【発明の名称】　情報処理方法および記録媒体
【請求項の数】　5
【発明者】
　【住所又は居所】　静岡県静岡市南町１８番１号　株式会社富士通静岡エンジニアリング内
　【氏名】　一瀬　克己
【発明者】
　【住所又は居所】　静岡県静岡市南町１８番１号　株式会社富士通静岡エンジニアリング内
　【氏名】　守屋　勝由
【特許出願人】
　【識別番号】　000005223
　【氏名又は名称】　富士通株式会社
【代理人】
　【識別番号】　100092152
　【弁理士】
　【氏名又は名称】　服部　毅巖
　【電話番号】　0426-45-6644
【手数料の表示】
　【予納台帳番号】　009874
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1
　【物件名】　図面　1

【物件名】　　要約書　1

【包括委任状番号】　9705176

【プルーフの要否】　　要

【書類名】　明細書

【発明の名称】　情報処理方法および記録媒体

【特許請求の範囲】

【請求項１】　複数のプロセッサを有する計算装置に対して、所定の情報処理を実行させる情報処理方法において、

実行対象となるプログラムを複数の並列処理ブロックに分割する並列処理ブロック分割ステップと、

前記並列処理ブロック分割ステップによって分割された並列処理ブロックを、前記複数のプロセッサのそれぞれに分担して処理させるための基本処理単位であるスレッドに分割するスレッド分割ステップと、

所定のプロセッサにおいて、前記スレッドの実行が終了した場合には、次の並列処理ブロックの実行を指示する指示ステップと、

を有することを特徴とする情報処理方法。

【請求項２】　前記指示ステップは、前記実行対象となるプログラムに所定の指示がなされている場合には、全てのスレッドの処理が終了するまで、次の並列処理ブロックの実行を指示しないことを特徴とする請求項１記載の情報処理方法。

【請求項３】　複数のプロセッサを有する計算装置に対して、所定の情報処理を実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体において、

コンピュータを、

実行対象となるプログラムを複数の並列処理ブロックに分割する並列処理ブロック分割手段、

前記並列処理ブロック分割手段によって分割された並列処理ブロックを、前記複数のプロセッサのそれぞれに分担して処理させるための基本処理単位であるスレッドに分割するスレッド分割手段、

所定のプロセッサにおいて、前記スレッドの実行が終了した場合には、次の並列処理ブロックの実行を指示する指示手段、

として機能させるプログラムを記録したコンピュータ読み取り可能な記録媒体

。

【請求項４】　複数のプロセッサを有する計算装置に対して、所定の情報処理を実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体において、

コンピュータを、

複数の並列処理ブロックに分割された実行対象のプログラムの所定の並列処理ブロックから実行要求がなされた場合には、複数のスレッドを生成し、各プロセッサに処理を分担させる処理分担手段、

何れかのスレッドの処理が終了した場合には、次の並列処理ブロックに係るスレッドの実行を指示する実行指示手段、

として機能させるプログラムを記録したコンピュータ読み取り可能な記録媒体

。

【請求項５】　前記実行指示手段は、前記実行対象となるプログラムに所定の指示がなされている場合には、全てのスレッドの処理が終了するまで、次の並列処理ブロックの実行を指示しないことを特徴とする請求項４記載の記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は情報処理方法および記録媒体に関し、特に複数のプロセッサを有する計算装置に対して、所定の情報処理を実行させる情報処理方法および記録媒体に関する。

【０００２】

【従来の技術】

複数のプロセッサを有する計算装置にプログラムを実行させる場合には、対象となるプログラムを複数の並列処理ブロックに分割し、得られた並列処理ブロックを、処理の基本単位であるスレッドに分割して各プロセッサに分担して処理させる方法が一般的であった。

【０００３】

図１０は、このような方法により図の左側に示すプログラムを５つのプロセッ

サを有する計算装置で実行する場合の様子を示す図である。この例では、実行対象となるプログラムには２つの処理ループ（行番号「１」～「３」および行番号「５」～「７」）が含まれており、それぞれが第１の並列処理ブロックおよび第２の並列処理ブロックとされている。

【０００４】

このプログラムが実行されると、各プロセッサに対応する処理の基本単位であるスレッド＃１～＃５が生成され、スレッド＃１は、最初の処理ループ（行番号「１」～「３」のループ）の変数ｉが１から２００までの処理を分担し、スレッド＃２は、同じく最初の処理ループの変数ｉが２０１から４００までの処理を分担する。同様にして、スレッド＃３、スレッド＃４、および、スレッド＃５は、それぞれ、４０１から６００、６０１から８００、および、８０１から１０００までの処理を分担する。

【０００５】

【発明が解決しようとする課題】

しかし、従来においては、並列処理ブロック間には「バリア」と呼ばれる監視機構が設けられており、全てのスレッドの処理が終了するまで次の並列処理ブロックの実行が保留されていた。

【０００６】

従って、複数の並列処理ブロックが存在する場合には、処理に要する時間は各並列処理ブロックにおいて最も遅いスレッドの処理時間を加算した時間となり、プロセッサ資源が有効に活用できない場合があるという問題点があった。

【０００７】

本発明は、以上のような点に鑑みてなされたものであり、プロセッサ資源を有効活用することにより、処理時間を短縮することが可能な情報処理方法を提供することを目的とする。

【０００８】

【課題を解決するための手段】

本発明では上記課題を解決するために、図１に示す、複数のプロセッサ（プロセッサ群４）を有する計算装置に対して、所定の情報処理を実行させる情報処理

方法において、実行対象となるプログラムを複数の並列処理ブロックに分割する並列処理ブロック分割ステップ１と、並列処理ブロック分割ステップ１によって分割された並列処理ブロックを、複数のプロセッサのそれぞれに分担して処理させるための基本処理単位であるスレッドに分割するスレッド分割ステップ２と、所定のプロセッサにおいて、スレッドの実行が終了した場合には、次の並列処理ブロックの実行を指示する指示ステップ３と、を有することを特徴とする情報処理方法が提供される。

【０００９】

ここで、並列処理ブロック分割ステップ１は、実行対象となるプログラムを複数の並列処理ブロックに分割する。スレッド分割ステップ２は、並列処理ブロック分割ステップ１によって分割された並列処理ブロックを、複数のプロセッサのそれぞれに分担して処理させるための基本処理単位であるスレッドに分割する。指示ステップ３は、所定のプロセッサにおいて、スレッドの実行が終了した場合には、次の並列処理ブロックの実行を指示する。

【００１０】

【発明の実施の形態】

以下、本発明の実施の形態を図面を参照して説明する。

図１は、本発明の情報処理方法の原理を説明する原理図である。この図に示すように、本発明に係る情報処理方法は、プロセッサ群４を有する情報処理装置に所定の情報処理を実行させることを目的としている。ここで、本発明に係る情報処理方法は、並列処理ブロック分割ステップ１、スレッド分割ステップ２、および、指示ステップを有している。

【００１１】

並列処理ブロック分割ステップ１は、実行対象となるプログラムを複数の並列処理ブロックに分割する。

スレッド分割ステップ２は、並列処理ブロック分割ステップ１によって分割された並列処理ブロックを、プロセッサ群４のそれぞれのプロセッサに分担して処理させるための基本処理単位であるスレッドに分割する。

【００１２】

指示ステップ３は、所定のプロセッサにおいて、スレッドの実行が終了した場合には、次の並列処理ブロックの実行を指示するが、プログラムに所定の指示がなされている場合には、全てのスレッドの処理が終了するまで、次の並列処理ブロックの実行を指示しない。

【００１３】

次に、以上の原理図の動作について説明する。

いま、実行対象のプログラムが入力されると、並列処理ブロック分割ステップ１は、これを複数の並列処理ブロックに分割する。なお、並列処理ブロックとは、例えば、ループ処理などのような一定の機能的な一体性を有する処理単位をいう。従って、プログラムに複数のループ処理が含まれている場合には、これらがそれぞれ並列処理ブロックとして分割される。

【００１４】

スレッド分割ステップ２は、並列処理ブロック分割ステップ１によって生成された並列処理ブロックのそれぞれを実行される順に取得し、各プロセッサに分担させるべき処理単位としてのスレッドに分割する。図１の例では、５つのプロセッサが存在しているので、並列処理ブロックは５つのスレッドに分割される。

【００１５】

プロセッサ群４は、スレッド分割ステップ２によって生成されたスレッドを、各プロセッサによって分担して処理する。このとき、プロセッサによって実行される処理の内容は、同一ではないので、処理が終了するタイミングは、プロセッサ毎に異なることになる。

【００１６】

従来においては、全てのプロセッサによる処理が完了するまで、バリアが監視していたので、あるプロセッサが先に処理を終了した場合であっても他の全てのプロセッサが処理を完了するまで待つ必要があった。

【００１７】

しかしながら、本実施の形態では、所定のプロセッサにおいて処理が完了した場合には、プログラムに指示がない限り、指示ステップ３が次の並列処理ブロックの実行を指示するので、先にスレッドの実行を終えたプロセッサは次の並列処

理ブロックのスレッドを実行することになる。

【００１８】

そして、実行処理が継続し、所定の指示がプログラムに出現した場合には、指示ステップ３が次の並列処理ブロックへの移行を保留するので、そこで、全てのスレッドが同期することになる。

【００１９】

以上に説明したように、従来では各並列処理ブロックにおける最長のスレッドの実行時間を加算した値が実行時間であったが、本発明の情報処理方法によれば、所定の並列ブロックに全ての最長のスレッドが分担されない限りは、実行時間を短縮することが可能となる。

【００２０】

次に、本発明の実施の形態について説明する。

図２は、本発明の情報処理方法を実行する実施の形態の構成例を示す図である。

【００２１】

本発明の情報処理方法は、図１に示すような情報処理装置において実行される。ここで、情報処理装置１０は、ＰＵ（Processor Unit）１０ａ－１～１０ａ－５、ＲＯＭ（Read Only Memory）１０ｂ、ＲＡＭ（Random Access Memory）１０ｃ、ＨＤＤ（Hard Disk Drive）１０ｄ、ＧＢ（Graphics Board）１０ｅ、Ｉ／Ｆ（Interface）１０ｆ、および、バス１０ｇによって構成されており、その外部には表示装置１１および入力装置１２が接続されている。

【００２２】

ここで、ＰＵ１０ａ－１～１０ａ－５は、ＨＤＤ１０ｄに格納されたプログラムに従って、各種演算処理を実行するとともに、装置の各部を制御する。

ＲＯＭ１０ｂは、ＰＵ１０ａ－１～１０ａ－５が実行する基本的なプログラムやデータ等を格納している。

【００２３】

ＲＡＭ１０ｃは、ＰＵ１０ａ－１～１０ａ－５が実行対象とするプログラムや演算途中のデータを一時的に格納する。

ＨＤＤ１０ｄは、ＰＵ１０ａ－１～１０ａ－５が実行するプログラムやデータを格納している。具体的には、システムを管理し、基本的なユーザ操作環境を提供するための基本的なプログラムであるＯＳ（Operating System）や、本発明に係るコンパイラ、リンカ、および、実行対象となるアプリケーションプログラム等を格納している。

【００２４】

ＧＢ１０ｅは、ＰＵ１０ａ－１～１０ａ－５から供給された描画命令に従って描画処理を施し、得られた画像を映像信号に変換して出力する。

Ｉ／Ｆ１０ｆは、入力装置１２から出力されたデータを、装置内部の表現形式に変換して入力する。

【００２５】

バス１０ｇは、ＰＵ１０ａ－１～１０ａ－５、ＲＯＭ１０ｂ、ＲＡＭ１０ｃ、ＨＤＤ１０ｄ、ＧＢ１０ｅ、および、Ｉ／Ｆ１０ｆを相互に接続し、これらの間で情報の授受を可能とする。

【００２６】

なお、ＰＵ１０ａ－１～ＰＵ１０ａ－５は、並行して処理を行うことが可能であり、実行対象のプログラムを複数の並列処理ブロックに分割して処理する。その際、共通のリソースであるＲＯＭ１０ｂ等にアクセスする際には、排他制御により、他のユニットの影響を受けないように制御されている。

【００２７】

また、以上の構成例は、説明を一部簡略化して示してあるが、要は複数のＰＵと、それぞれのＰＵが独立して動作するような構成を有していればよい。

図３は、図２に示す実施の形態において、ＨＤＤ１０ｄに格納されているＯＳが起動され、そのＯＳ上で、本発明に係るコンパイラ、リンカ、および、ライブラリが実行される際のそれぞれの対応関係を示す図である。この図に示すように、実行対象となるソースプログラム２０は、コンパイラ２１によってコンパイル（翻訳）され、リンカ２２によって基本的なプログラムであるライブラリ２３に格納された必要なプログラムが付加され、実行形式プログラム２４が生成される。なお、本発明は、ライブラリ２３の詳細に係るものであり、以下に示す新たな

制御方法を用いることにより、並列処理の高速化を図ることが可能となる。

【００２８】

図４は、図３に示すソースプログラム２０の一例を示す図である。

このソースプログラム２０では、行番号「１」において、要素数が１０００，２０００，３０００である整数型の配列ｍ，ｎ，ｐが宣言されている。行番号「３」においては並列処理を行うことが宣言されており、これは行番号「１９」に示す「ＥＮＤ」と対を構成している。行番号「４」から「８」まではループ処理を形成しており、変数ｉの値を配列ｍの各要素として格納する処理が実行される。また、行番号「９」から「１３」までは他のループ処理を形成しており、変数ｉを２倍した値を配列ｎの各要素として格納する処理が実行される。更に、行番号「１４」から「１８」までは他のループ処理を形成しており、変数ｉを３倍した値を配列ｐの各要素として格納する処理が実行される。

【００２９】

以下では、行番号「４」から「８」までを第１のループ処理、行番号「９」から「１３」までを第２のループ処理、行番号「１４」から「１８」までを第３のループ処理と呼ぶことにする。行番号「８」，「１３」，「１８」に示す「ＮＯＷＡＩＴ」は、ウエイト処理を実行しないことを示しており、この一行を付加することにより、バリア機能が停止されることになる。

【００３０】

なお、処理の並列数は、図示せぬ初期設定用のプログラムによって指定することができる。また、プログラム中においても、所定のコマンドを配置することにより、並列数を任意に指定することができる。

【００３１】

以上のようなソースプログラム２０は、コンパイラ２１によって翻訳され、リンカ２２によってライブラリ２３に含まれる必要なプログラムが付加され、実行形式プログラム２４に変換される。なお、コンパイルの際には、並列処理の単位である並列処理ブロックに分割され、それぞれの並列処理ブロックにはユニークな番号である並列処理ブロック番号が付与される。図４に示すソースプログラムの例では、第１～第３のループ処理のそれぞれが並列処理ブロックであるので、

例えば、１～３の並列ブロック番号がそれぞれの並列処理ブロックに付与される。

【００３２】

図５は、実行形式プログラム２４が実行された際の動作を説明するための図である。この図において、実行形式プログラム２４は、図４に示すソースプログラム２０が機械語に翻訳されたものであり、前述した第１～第３のループ処理を有している。スレッド制御部６０および第１～第５のスレッド６１～６５は、リンカ２２によって付加された並列処理用のプログラムであり、実行形式プログラム２４から呼び出されて実行されるサブルーチン形式のプログラムである。

【００３３】

ここで、スレッド制御部６０は、並列処理ブロックである第１～第３のループ処理から呼び出された場合には、第１～第５のスレッド６１～６５を起動し、処理を実行させるとともに、これらのスレッドの実行状態の管理を行う。第１～第５のスレッドは、ＰＵ１０ａ－１～１０ａ－５によって実行される処理の基本単位である。

【００３４】

図６は、図５に示すスレッド制御部６０および第１～第５のスレッド６１～６５が機能する際にＲＡＭ１０ｃに確保する記憶領域を示す図である。この図において、スレッド情報域７１～７５は第１～第５のスレッド６１～６５のそれぞれが確保している領域であり、各情報域にはそのスレッドが現在実行している並列処理ブロックの番号を示す並列処理ブロック番号７１ａ～７５ａが格納されている。並列処理制御情報域７６は、スレッド制御部６０が確保している領域であり、全てのスレッドのうち最先のスレッドが実行している並列処理ブロック番号７６ａが格納されている。

【００３５】

並列処理ブロック制御情報域７７，７８は、現在実行中の並列処理ブロックに対応して生成される情報域であり、当該並列処理ブロックの処理を終了したスレッドの個数である実行終了スレッド数７７ａ，７８ａが格納されている。この例では、２つの並列処理ブロック制御情報域７７，７８が生成されており、２つの

並列処理ブロックが実行中の状態に対応している。なお、この例では、第１のスレッド６１～第４のスレッド６４は、並列処理ブロック制御情報域７７に対応する並列処理ブロックを実行中であり、第５のスレッド６５は、並列処理ブロック制御情報域７８に対応する並列処理ブロックを実行中である。

【００３６】

なお、３以上の並列処理ブロックが実行中である場合にはその数に対応した並列処理ブロック制御情報域が生成され、また、１つの並列ブロックが実行中である場合には１つの並列処理ブロック制御情報域が生成される。

【００３７】

次に、図７を参照して、図４に示すソースプログラム２０から生成された実行形式プログラム２４が実行される際の、図５および図６に示すブロック図の動作について説明する。

【００３８】

実行形式プログラム２４が起動されると、並列処理ブロック番号が「１」である第１のループ処理が実行の対象となる。第１のループ処理は、スレッド制御部６０をサブルーチンコールし、引数等を引き渡すとともに、処理の開始を依頼する。すると、スレッド制御部６０は、並列処理制御情報域７６の並列処理ブロック番号を「１」に設定するとともに、並列処理ブロック制御情報域７７を生成し、実行終了スレッド数７７ａを「０」に初期設定する。続いて、スレッド制御部６０は、第１～第５のスレッドを生成して、処理の実行を開始する。第１～第５のスレッド６１～６５は、それぞれ、スレッド情報域７１～７５を確保するとともに、現在実行中である第１のループ処理に対応する並列処理ブロック番号「１」を並列処理ブロック番号７１ａ～７５ａとして格納する。

【００３９】

図７は、スレッド情報域７１～７５と、並列処理制御情報域７６に格納されているデータを示す図であり、左端の第１列目から第５列目までがスレッド情報域７１～７５にそれぞれ対応し、右端の列が並列処理制御情報域７６に対応している。この図の第１行目は、処理が開始された当初の状態を示しており、開始当初は第１のスレッド６１のみが並列処理ブロック番号「１」を格納しておりその他

はまだ未格納の状態である。第２行目では、５列目以外は全て「１」の状態となっており、第５のスレッド６５以外は全て第１のループ処理を実行中の状態であることが示されている。

【００４０】

このような状態において、第２のスレッド６２が第１のループ処理の実行を終了すると、スレッド制御部６０にその旨が通知される。すると、スレッド制御部６０は、第１の処理ループの末尾に「ＮＯＷＡＩＴ」が挿入されていることを検出し、バリア機能を実行せずに次の処理に移行することを認識し、先ず、実行終了スレッド数７７ａを「０」から「１」に更新する。次に、スレッド制御部６０は、並列処理制御情報域７６の並列処理ブロック番号７６ａを「１」から「２」に更新し、続いて、第２のループ処理に対応した並列処理ブロック制御情報域７８を生成して実行終了スレッド数７８ａを「０」に初期設定する。

【００４１】

以上のような動作が繰り返され、全てのスレッドが第１のループ処理を終了した場合には、実行終了スレッド数７７ａが「５」になるので、その場合には、並列処理ブロック制御情報域７７がＲＡＭ１０ｃ上から削除されることになる。

【００４２】

そして、全てのスレッドが第３のループ処理を終了した場合には、図７の最後の行に示すように、並列処理ブロック番号７１ａ～７５ａが全て「３」の状態になり、プログラムの実行が完了することになる。

【００４３】

従って、以上の実施の形態では、図１０に示す従来例とは異なり、図８に示すようにバリア機能が実行されることなく、各スレッドが他のスレッドの実行状態に拘わりなく、次の並列処理ブロックを実行することになる。従って、図１０に示す従来例では、トータルの実行時間は各並列処理ブロックで最も遅いスレッドの実行時間の合計になるが、本実施の形態ではそれ以下の時間で実行することが可能となる。

【００４４】

なお、以上の実施の形態では、５つのＰＵ１０ａ－１～１０ａ－５により、５

つのスレッドで処理する場合を例に挙げて説明したが、本発明はこのような場合に限定されるものではなく、これ以外の組み合わせでも本発明を適用可能であることはいうまでもない。

【００４５】

また、以上の実施の形態では、スレッドの管理をライブラリ２３によって行うようにしたが、コンパイラ２１によって同様のプログラムを実行形式プログラム２４に付加するようにしても同様の効果を得ることができる。なお、付加の方法としては、例えば、インライン展開やマクロ等を用いることができる。

【００４６】

更に、以上の実施の形態では、全ての並列処理ブロックの最後に「ＮＯＷＡＩＴ」を挿入したので、各スレッドは、他のスレッドとは無関係に最後の並列処理ブロックまで処理を継続することになるが、この「ＮＯＷＡＩＴ」を除外することにより、そこで、スレッドの足並みを一旦揃えることも可能である。このように「ＮＯＷＡＩＴ」を適宜挿入することにより、バリア機能を有効または無効にすることが可能となる。

【００４７】

続いて、図９を参照し、スレッド制御部６０において実行されるフローチャートについて説明する。このフローチャートは、第１～第５のスレッド６１～６５において処理が終了した場合に呼び出されて実行される。このフローチャートが開始されると、以下のステップが実行される。

【００４８】

ステップＳ１０：

該当するスレッド情報域の並列処理ブロック番号を１だけインクリメントする。

【００４９】

ステップＳ１１：

変数ｊに並列処理制御情報域の並列処理ブロック番号を代入する。

ステップＳ１２：

変数ｋに該当するスレッド情報域の並列処理ブロック番号を代入する。

【００５０】

ステップＳ１３：

変数ｊの値が変数ｋの値以上であるか否か、即ち、当該スレッドが新たな並列処理ブロックを実行するか否かを判定し、新たな並列処理ブロックを実行する場合にはステップＳ１４に進み、それ以外の場合にはステップＳ１７に進む。

【００５１】

ステップＳ１４：

並列処理制御情報域７６の並列処理ブロック番号７６ａを１だけインクリメントする。

【００５２】

ステップＳ１５：

新たな並列処理ブロックに対応する並列処理ブロック制御情報域を生成するとともに、実行終了スレッド数を「０」に初期設定する。

【００５３】

ステップＳ１６：

新たなスレッドの実行処理を開始する。

ステップＳ１７：

スレッド情報域の並列処理ブロック番号を参照し、実行対象となる並列処理ブロックを特定する。

【００５４】

ステップＳ１８：

ステップＳ１７で特定した並列処理ブロックを実行する。

ステップＳ１９：

該当する並列処理ブロック制御情報域の実行終了スレッド数をインクリメントする。

【００５５】

ステップＳ２０：

ステップＳ１９におけるインクリメントの結果、実行終了スレッド数がスレッド数（図６の例では「５」）と等しくなった場合には、ステップＳ２１に進み、

それ以外の場合には処理を終了する。

【００５６】

ステップＳ２１：

処理が終了した並列処理ブロック制御情報域を削除する。

以上の処理によれば、前述した機能を実現することが可能となる。

【００５７】

最後に、上記の処理機能は、コンピュータによって実現することができる。その場合、情報処理装置が有すべき機能の処理内容は、コンピュータで読み取り可能な記録媒体に記録されたプログラムに記述されており、このプログラムをコンピュータで実行することにより、上記処理がコンピュータで実現される。コンピュータで読み取り可能な記録媒体としては、磁気記録装置や半導体メモリ等がある。市場へ流通させる場合には、ＣＤ－ＲＯＭ(Compact Disk Read Only Memory)やフロッピーディスク等の可搬型記録媒体にプログラムを格納して流通させたり、ネットワークを介して接続されたコンピュータの記憶装置に格納しておき、ネットワークを通じて他のコンピュータに転送することもできる。コンピュータで実行する際には、コンピュータ内のハードディスク装置等にプログラムを格納しておき、メインメモリにロードして実行する。

【００５８】

（付記１）　複数のプロセッサを有する計算装置に対して、所定の情報処理を実行させる情報処理方法において、

実行対象となるプログラムを複数の並列処理ブロックに分割する並列処理ブロック分割ステップと、

前記並列処理ブロック分割ステップによって分割された並列処理ブロックを、前記複数のプロセッサのそれぞれに分担して処理させるための基本処理単位であるスレッドに分割するスレッド分割ステップと、

所定のプロセッサにおいて、前記スレッドの実行が終了した場合には、次の並列処理ブロックの実行を指示する指示ステップと、

を有することを特徴とする情報処理方法。

【００５９】

（付記２）　前記指示ステップは、前記実行対象となるプログラムに所定の指示がなされている場合には、全てのスレッドの処理が終了するまで、次の並列処理ブロックの実行を指示しないことを特徴とする付記１記載の情報処理方法。

【００６０】

（付記３）　複数のプロセッサを有する計算装置に対して、所定の情報処理を実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体において、

コンピュータを、

実行対象となるプログラムを複数の並列処理ブロックに分割する並列処理ブロック分割手段、

前記並列処理ブロック分割手段によって分割された並列処理ブロックを、前記複数のプロセッサのそれぞれに分担して処理させるための基本処理単位であるスレッドに分割するスレッド分割手段、

所定のプロセッサにおいて、前記スレッドの実行が終了した場合には、次の並列処理ブロックの実行を指示する指示手段、

として機能させるプログラムを記録したコンピュータ読み取り可能な記録媒体。

【００６１】

（付記４）　複数のプロセッサを有し、所定の情報処理を実行する情報処理装置において、

実行対象となるプログラムを複数の並列処理ブロックに分割する並列処理ブロック分割手段と、

前記並列処理ブロック分割手段によって分割された並列処理ブロックを、前記複数のプロセッサのそれぞれに分担して処理させるための基本処理単位であるスレッドに分割するスレッド分割手段と、

所定のプロセッサにおいて、前記スレッドの実行が終了した場合には、次の並列処理ブロックの実行を指示する指示手段と、

を有することを特徴とする情報処理装置。

【００６２】

（付記５）　複数のプロセッサを有する計算装置に対して、所定の情報処理を実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体において、

コンピュータを、

複数の並列処理ブロックに分割された実行対象のプログラムの所定の並列処理ブロックから実行要求がなされた場合には、複数のスレッドを生成し、各プロセッサに処理を分担させる処理分担手段、

何れかのスレッドの処理が終了した場合には、次の並列処理ブロックに係るスレッドの実行を指示する実行指示手段、

として機能させるプログラムを記録したコンピュータ読み取り可能な記録媒体。

【００６３】

（付記６）　前記実行指示手段は、前記実行対象となるプログラムに所定の指示がなされている場合には、全てのスレッドの処理が終了するまで、次の並列処理ブロックの実行を指示しないことを特徴とする付記５記載の記録媒体。

【００６４】

【発明の効果】

以上説明したように本発明では、複数のプロセッサを有する計算装置に対して、所定の情報処理を実行させる情報処理方法において、実行対象となるプログラムを複数の並列処理ブロックに分割する並列処理ブロック分割ステップと、並列処理ブロック分割ステップによって分割された並列処理ブロックを、複数のプロセッサのそれぞれに分担して処理させるための基本処理単位であるスレッドに分割するスレッド分割ステップと、所定のプロセッサにおいて、スレッドの実行が終了した場合には、次の並列処理ブロックの実行を指示する指示ステップと、を設けるようにしたので、並列処理ブロックが連続して処理される場合には、従来のようなバリアによる遅延を排除することにより、処理速度を向上させることが可能となる。

【図面の簡単な説明】

【図１】

本発明の動作原理を説明する原理図である。

【図２】

本発明の実施の形態の構成例を示すブロック図である。

【図３】

図２に示す実施の形態において、ＨＤＤに格納されているＯＳが起動され、そのＯＳ上で、本発明に係るコンパイラ、リンカ、および、ライブラリが実行される際のそれぞれの対応関係を示す図である。

【図４】

図３に示すソースプログラムの一例を示す図である。

【図５】

実行形式プログラムが実行された際の動作を説明するための図である。

【図６】

図５に示すスレッド制御部および第１～第５のスレッドが機能する際にＲＡＭに確保している記憶領域を示す図である。

【図７】

図４に示すソースプログラムから生成された実行形式プログラムが実行される際の、図５および図６に示すブロック図の動作について説明する図である。

【図８】

本実施の形態におけるスレッドの実行形態を説明するための図である。

【図９】

図５に示すスレッド制御部において実行される処理の一例を説明するフローチャートである。

【図１０】

従来例におけるスレッドの実行形態を説明するための図である。

【符号の説明】

１　並列処理ブロック分割ステップ

２　スレッド分割ステップ

３　指示ステップ

４　プロセッサ群

１０ａ－１～１０ａ－５　ＰＵ

１０ｂ　ＲＯＭ

１０ｃ　ＲＡＭ

１０ｄ　ＨＤＤ

１０ｅ　ＧＢ

１０ｆ　Ｉ／Ｆ

１０ｇ　バス

１１　表示装置

１２　入力装置

２０　ソースプログラム

２１　コンパイラ

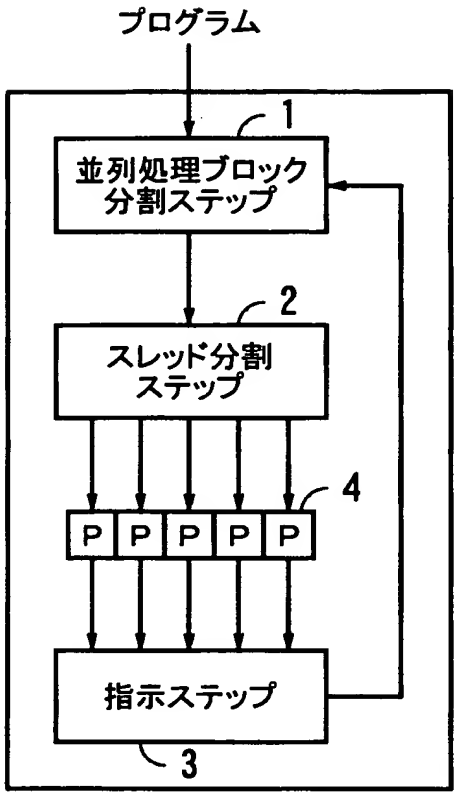
２２　リンカ

２３　ライブラリ

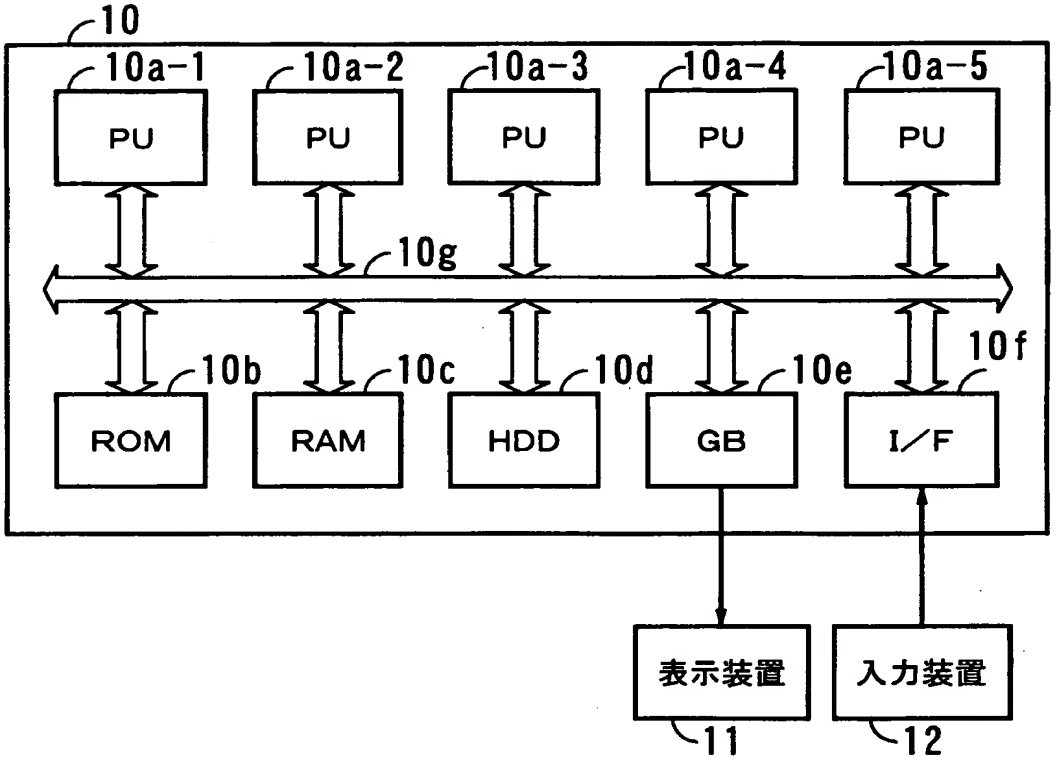
２４　実行形式プログラム

【書類名】　　図面

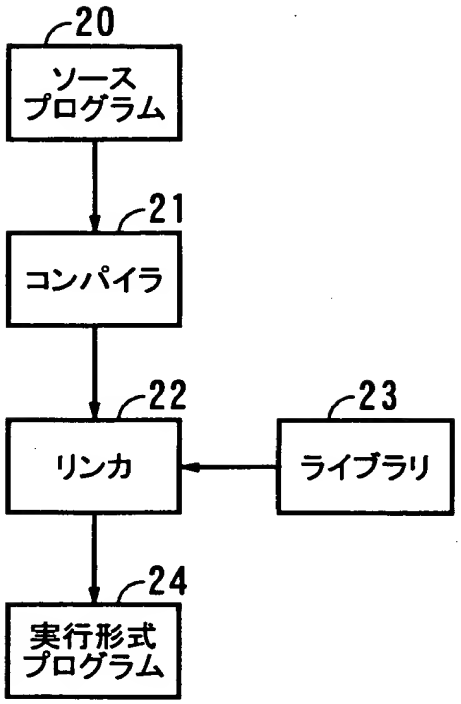
【図１】



【図２】


10
10a-1
10a-2
10a-3
10a-4
10a-5
PU
PU
PU
PU
PU
10g
10b
10c
10d
10e
10f
ROM
RAM
HDD
GB
I／F
表示装置
入力装置
11
12

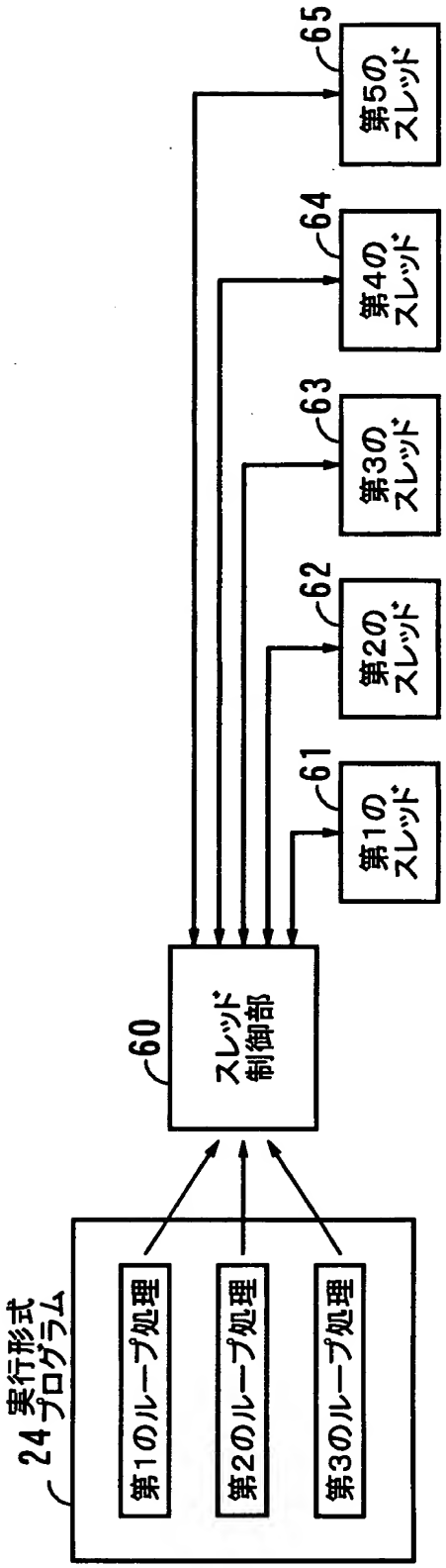
【図３】



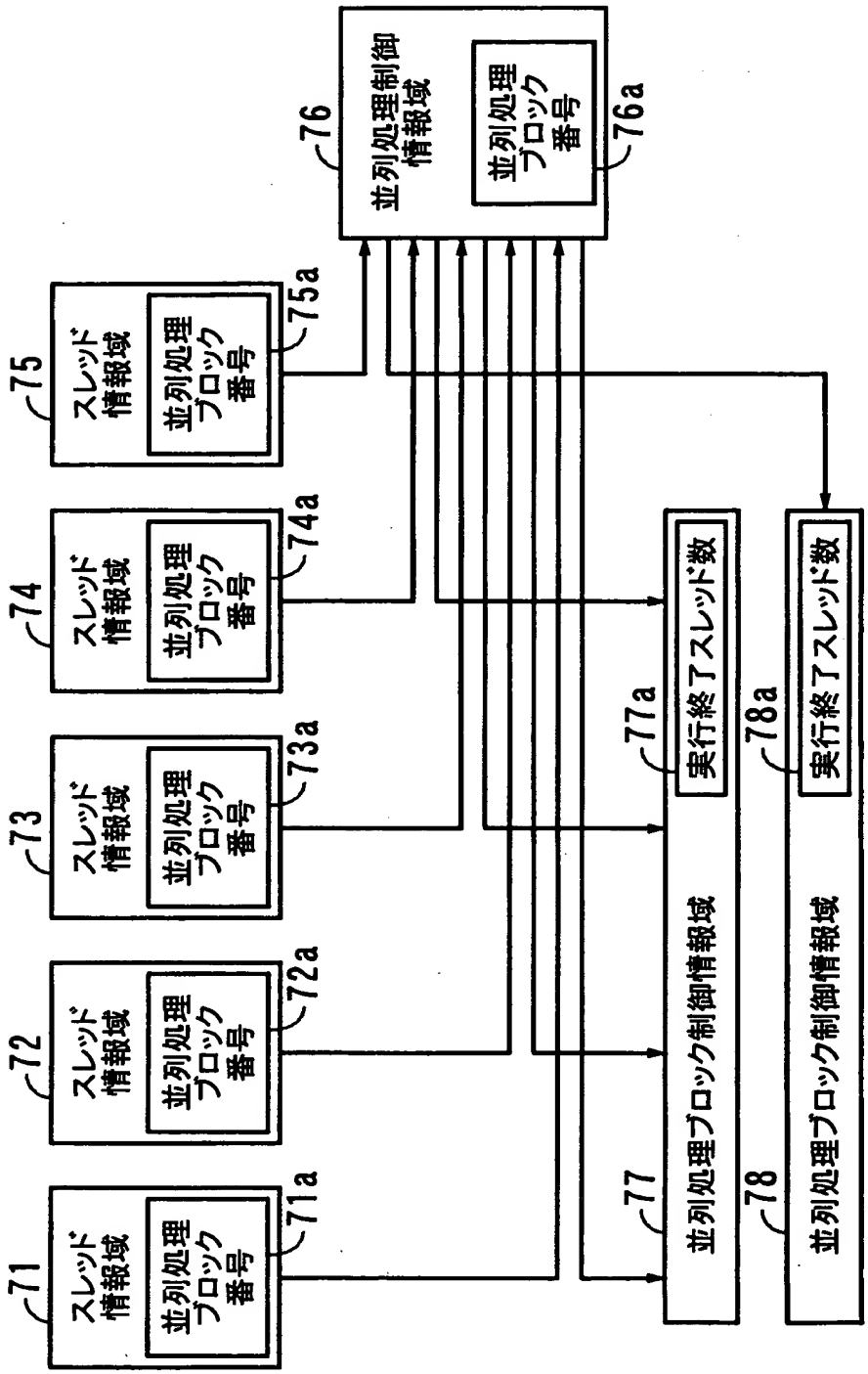
【図４】

```
1    INTEGER m(1000),n(2000),p(3000)
2
3    !$OMP PARALLEL
4          !$OMP DO
5               DO i=1,1000
6                    m(i)=i
7               END DO
8          !$OMP END DO NOWAIT
9          !$OMP DO
10              DO i=1,2000
11                   n(i)=i*2
12              END DO
13         !$OMP END DO NOWAIT
14         !$OMP DO
15              DO i=1,3000
16                   p(i)=i*3
17              END DO
18         !$OMP END DO NOWAIT
19   !$OMP END PARALLEL
20   END
```

【図５】


24 実行形式プログラム
第1のループ処理
第2のループ処理
第3のループ処理
60 スレッド制御部
61 第1のスレッド
62 第2のスレッド
63 第3のスレッド
64 第4のスレッド
65 第5のスレッド

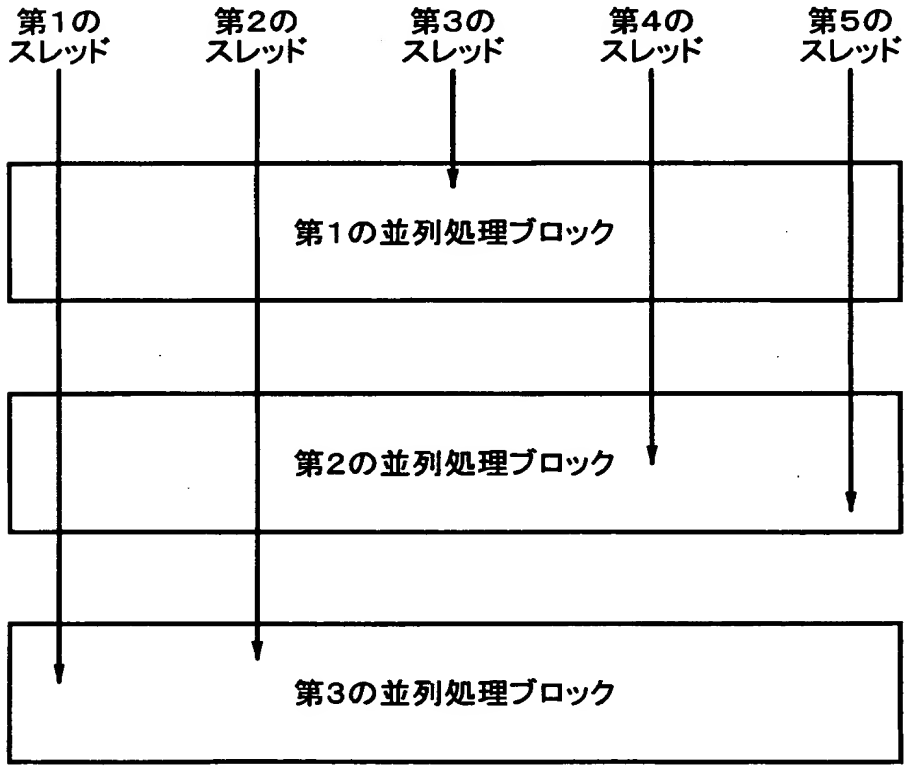
【図６】


71
スレッド情報域
並列処理ブロック番号
71a
72
スレッド情報域
並列処理ブロック番号
72a
73
スレッド情報域
並列処理ブロック番号
73a
74
スレッド情報域
並列処理ブロック番号
74a
75
スレッド情報域
並列処理ブロック番号
75a
76
並列処理制御情報域
並列処理ブロック番号
76a
77
並列処理ブロック制御情報域
実行終了スレッド数
77a
78
並列処理ブロック制御情報域
実行終了スレッド数
78a

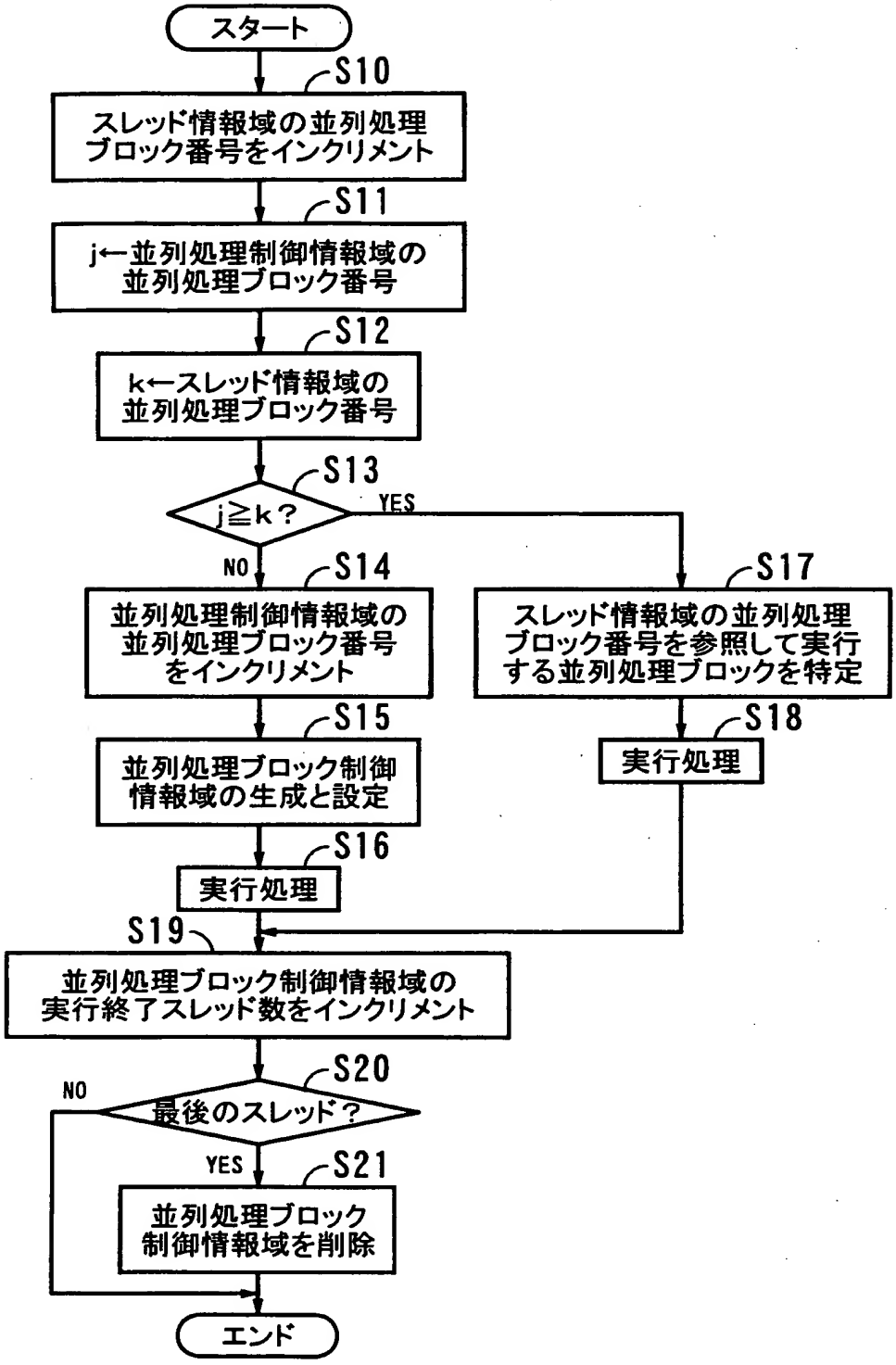
【図７】

第1のスレッド	第2のスレッド	第3のスレッド	第4のスレッド	第5のスレッド	先頭
1	−	−	−	−	1
1	1	1	1	−	1
1	2	1	1	1	2
⋮	⋮	⋮	⋮	⋮	⋮
3	2	2	2	2	3
3	3	3	3	3	3

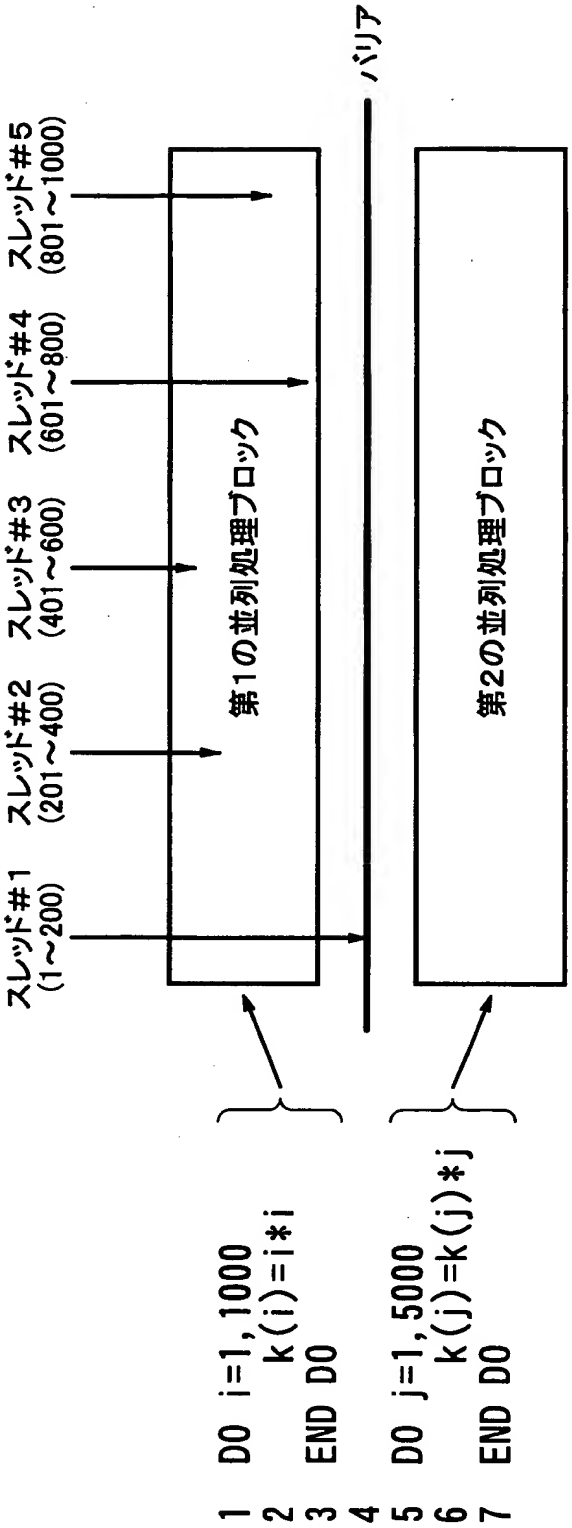
【図８】



【図９】


スタート
S10
スレッド情報域の並列処理ブロック番号をインクリメント
S11
j←並列処理制御情報域の並列処理ブロック番号
S12
k←スレッド情報域の並列処理ブロック番号
S13
j≧k？
YES
NO
S14
並列処理制御情報域の並列処理ブロック番号をインクリメント
S15
並列処理ブロック制御情報域の生成と設定
S16
実行処理
S17
スレッド情報域の並列処理ブロック番号を参照して実行する並列処理ブロックを特定
S18
実行処理
S19
並列処理ブロック制御情報域の実行終了スレッド数をインクリメント
S20
最後のスレッド？
NO
YES
S21
並列処理ブロック制御情報域を削除
エンド

【図１０】


スレッド#1 (1～200)
スレッド#2 (201～400)
スレッド#3 (401～600)
スレッド#4 (601～800)
スレッド#5 (801～1000)
第1の並列処理ブロック
バリア
第2の並列処理ブロック
1 DO i=1,1000
k(i)=i*i
2
3 END DO
4
5 DO j=1,5000
k(j)=k(j)*j
6
7 END DO

【書類名】　　要約書

【要約】

【課題】　並列処理計算装置の実行速度を向上させる。

【解決手段】　並列処理ブロック分割ステップ１は、実行対象となるプログラムを複数の並列処理ブロックに分割する。スレッド分割ステップ２は、並列処理ブロック分割ステップ１によって生成された並列処理ブロックを、プロセッサ群４の個数に応じた複数のスレッドに分割する。プロセッサ群４は、スレッド分割ステップ２によって生成されたスレッドを実行する。指示ステップ３は、プロセッサ群４のそれぞれのプロセッサがスレッドの実行を終了した場合であって、プログラムに所定の指示がなされていない場合には、当該プロセッサに対して次の並列処理ブロックに係るスレッドを実行するように指示する。

【選択図】　　図１

出　願　人　履　歴　情　報

識別番号　　〔０００００５２２３〕

１．変更年月日　　１９９６年　３月２６日
［変更理由］　住所変更
住　所　　神奈川県川崎市中原区上小田中４丁目１番１号
氏　名　　富士通株式会社